Towards “Chemical” Grids

Institute of Computing Technology
Research Center for Grid and Service Computing
Yann Radenac

yann.radenac@software.ict.ac.cn
Outline

1. Chemical Programming
2. Chemical Grid Programming
3. Perspectives
1. Chemical Programming

2. Chemical Grid Programming

3. Perspectives
Chemical Programming

- Initial work from Jean-Pierre Banâtre and Daniel Le Métayer (1986)
- Programming model using chemistry as a metaphor:
  - data = molecules
  - computation = chemical reactions
- Last development: HOCL
Higher Order Chemical Language

- HOCL: Higher Order Chemical Language
- Based on the $\gamma$-calculus
- A HOCL program is a chemical solution of atoms $\langle A_1, \ldots, A_n \rangle$
- Atoms $A_i$ may be:
  - Integers, strings, ... any external object
  - Tuples $A_1: \ldots : A_k$
  - Sub-solutions
  - One-shot rules: one $P$ by $M$ if $C$
  - N-shot rules: replace $P$ by $M$ if $C$
Example: computing the prime numbers lower than 10
Example: computing the prime numbers lower than 10
Example: computing the prime numbers lower than 10

- sieve = replace x; y by x if x div y
Example: computing the prime numbers lower than 10

where $sieve = replace x, y$ by $x$ if $x$ div $y$
Example: computing the prime numbers lower than 10

where sieve = replace x, y by x if x div y
Example: computing the prime numbers lower than 10

where \( \text{sieve} = \text{replace } x, y \text{ by } x \text{ if } x \text{ div } y \)
Example: computing the prime numbers lower than 10

where \( \text{sieve} = \text{replace} \, x, \, y \, \text{by} \, x \, \text{if} \, x \, \text{div} \, y \)
Example: computing the prime numbers lower than 10

where \( \text{sieve} = \text{replace } x, y \text{ by } x \text{ if } x \text{ div } y \)
Example: computing the prime numbers lower than 10

where \( \text{sieve} = \text{replace } x, y \text{ by } x \text{ if } x \div y \)
Example: computing the prime numbers lower than 10

where sieve = replace x, y by x if x div y
Outline

1. Chemical Programming

2. Chemical Grid Programming

3. Perspectives
Main reasons:

- Conventional approaches seem to fail
Main reasons:

– Conventional approaches seem to fail
– Very high level parallel language: no process, provides atomicity and mutual exclusion, shared memory, etc.
Main reasons:

– Conventional approaches seem to fail
– Very high level parallel language: no process, provides atomicity and mutual exclusion, shared memory, etc.
– No central control (locality property)
Main reasons:

- Conventional approaches seem to fail
- Very high level parallel language: no process, provides atomicity and mutual exclusion, shared memory, etc.
- No central control (locality property)
- Chemical metaphor
Main reasons:

- Conventional approaches seem to fail
- Very high level parallel language: no process, provides atomicity and mutual exclusion, shared memory, etc.
- No central control (locality property)
- Chemical metaphor
- Autonomic programs...
Autonomic property

\[ \langle \text{sieve}, 2, 3, 4, 5, 6, 7, 8, 9, 10 \rangle \]

\[ \downarrow \ast \]

\[ \langle \text{sieve}, 2, 3, 5, 7 \rangle \]
Autonomic property

\[ \langle \text{sieve}, 2, 3, 4, 5, 6, 7, 8, 9, 10 \rangle \]

**Stabilizing**

\[ \downarrow \ast \]

\[ \langle \text{sieve}, 2, 3, 5, 7 \rangle \]

**Perturbation**

\[ \langle \text{sieve}, 2, 3, 5, 7, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20 \rangle \]
Autonomic property

\[\langle \text{sieve}, 2, 3, 4, 5, 6, 7, 8, 9, 10 \rangle\]

**Stabilizing**

\[\downarrow \ast\]

\[\langle \text{sieve}, 2, 3, 5, 7 \rangle\]

**Perturbation**

\[\langle \text{sieve}, 2, 3, 5, 7, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20 \rangle\]

**Re-stabilizing**

\[\downarrow \ast\]

\[\langle \text{sieve}, 2, 3, 5, 7, 11, 13, 17, 19 \rangle\]
Chemical Grid Programming

Application level: write applications with HOCL

System level: specification and interface of “chemical” grids with HOCL
Application level

- Applications are chemical programs
- No reference to any grid mechanism
- HOCL used as a coordination language
Appl. level: Ray Tracing Example

observer

screen

initial ray

shadow ray

reflected ray

transmitted ray

light source

transparent object

opaque object
Appl. level: Ray Tracing Example

```python
import Pixel, Ray, LightRay, Contrib, Scene.
```
import Pixel, Ray, LightRay, Contrib, Scene.

\langle p_1, \ldots, p_n, \text{renderPixel}\rangle
import Pixel, Ray, LightRay, Contrib, Scene.

\[\langle p_1, \ldots, p_n, \text{renderPixel}\rangle\]

renderPixel = replace \(p::\text{Pixel}\)

by \(\langle \text{firstRay}(p), \text{renderRay}, \text{deleteRay}, \text{enlighten}, \text{sumContrib}\rangle\)
import Pixel, Ray, LightRay, Contrib, Scene.

renderPixel = replace p::Pixel
            by \langle firstRay(p), renderRay, deleteRay, enlighten, sumContrib \rangle

renderRay = replace r::Ray
            by Scene.intersectRays(r)
            if r.contribution() > \epsilon

deleteRay = replace r::Ray, \omega by \omega if r.contribution() \leq \epsilon

enlighten = replace l::LightRay
            by l.computeContrib()

sumContrib = replace c_1::Contrib, c_2::Contrib
            by c_1.add(c_2)
System level

- Grid viewed as a chemical solution:
  - resources = molecules
  - coordination = chemical reactions

- Chemical program as:
  - a specification
  - an interface for the grid administrator

Systems level: Chemical Grid Ray Tracing

\[ \langle R_1:\langle r_1 \rangle, \ldots, \langle r_n \rangle \rangle, \ R_2:\emptyset, \ldots, \ R_n:\emptyset, \ renderRules, \ split, \ merge, \ newRes, \ remRes \]
Sys. level: Chemical Grid Ray Tracing

\[
\langle R_1: \langle r_1 \rangle, \ldots, \langle r_n \rangle \rangle, \quad R_2: \langle \rangle, \ldots, \quad R_n: \langle \rangle, \quad \text{renderRules, split, merge, newRes, remRes}\rangle
\]

\[
\text{split} = \text{replace } res_1: \langle r::\text{Ray, } \omega_p \rangle, \omega_1 \rangle, \quad res_2: \langle \omega_2 \rangle \\
\text{by } res_1: \langle \omega_1 \rangle, \quad res_2: \langle r, \omega_p \rangle, \omega_2 \rangle \\
\text{if } res_1.\text{load()} \gg res_2.\text{load()}
\langle R_1: \langle \langle r_1 \rangle, \ldots, \langle r_n \rangle \rangle, \ldots, R_n: \langle \rangle \rangle, \ renderRules, \\
\text{split, merge, newRes, remRes} \rangle

\text{split} = \text{replace } \langle r::\text{Ray}, \omega_p \rangle, \omega_1, \langle \omega_2 \rangle \\
\text{by } \langle \omega_1 \rangle, \langle r, \omega_p \rangle, \omega_2 \\
\text{if } \text{res}_1.\text{load}() \gg \text{res}_2.\text{load}()

\text{merge} = \text{replace } \langle \omega_1 \rangle, \langle c::\text{Contrib} \rangle, \omega_2 \\
\text{by } \langle \omega_1, \langle c \rangle \rangle, \langle \omega_2 \rangle \\
\text{if } \text{res}_1 < \text{res}_2
Outline

1. Chemical Programming

2. Chemical Grid Programming

3. Perspectives
Currently EchoGrid Fellow at the ICT

Initial work:

Application level: implement HOCL using GSML
Thank You!